

**Robots & Spiders & Crawlers:
How Web and intranet search engines
follow links to build indexes**

A white paper

By Avi Rappaport, Search Tools Consulting



BEST AVAILABLE COPY

Purpose

This white paper is intended to give general information regarding the way search engines collect information to fill the search index. Also discussed are potential problems that spiders may have with certain types of content and how to overcome those problems. Organization notes:

- Shaded boxes within the sections explain the way Ultraseek Server 3.1 handles the topic discussed
- Exhibits mentioned with the text are located in Section XI
- The author frequently refers to other sites; links to those sites are in Section X

This paper was originally published in October 1999 online at <http://software.infoseek.com>.

About the Author

Avi Rappoport is the founder of Search Tools Consulting, providing analysis and advice on Web site, Intranet and portal search engines. She also analyzes the search tools industry and information retrieval issues, and publishes reports at www.searchtools.com. Avi is a librarian who has spent ten years in software design, implementation and management, and is pleased to have found a way to combine both areas of expertise. You can reach her at consult@searchtools.com.

Table of Contents

I.	Introduction	3
II.	How Robots Follow Links to Find Pages	3
III.	Communicating With Robots	7
IV.	Indexing Process	9
V.	Details of Following Links and Crawling Sites	11
VI.	Dealing with Dynamic Data	21
VII.	Detecting Duplicate Pages	26
VIII.	Revisiting Sites & Updating Indexes	27
IX.	Conclusion	28
X.	Useful Robot Sites	29
XI.	Exhibits	30

I. Introducing Search Engine Robots & Spiders

Search engine robots are programs that follow links on Web sites, gathering data for search engine indexes. They are considered *robots* rather than programs because they are somewhat independent: no programmer has told them what pages to find. Other names for these programs are *spider* or *crawler* (because they follow links on the Web), *worm*, *wanderer*, *gatherer*, and so on.

Web-wide search engines, such as Go Network, AltaVista and HotBot use robots because they are accessing the pages remotely, just as a person browsing a site would view the pages in their browser. Intranet and site search engines, such as Ultraseek Server, can also use robots to gather the data for indexing. All of these search robots behave in essentially similar ways, so designing links for one robot to crawl will likely improve your compatibility with the other robots as well.

In this paper, you will find call-outs such as this to tell you how Ultraseek Server handles the topics in that section.

You may be wondering why the robots crawl your site at all -- why not just search the files when someone types a word and hits a button? The reason is efficiency: thirty years of Information Retrieval research have found that storing data in indexes reduces the load on the server, allows very large collections to be searched by many people at the same time, and lets the search engine list the pages in order by the relevance of their text to the search terms.

Note: The other way search indexers may gather data is to read files from specified folders. This is faster than using a robot, and it can use the file system information to check whether a file has been changed since the last index. However, robots can access multiple servers without having to mount volumes on a network, and they are also less likely to index private or obsolete data, because they follow links rather than indexing everything in the folders.

This paper explains how robots gather data for search engines, how they follow links, how to control robots indexing and updating, and more. If you're running a search engine indexing robot, you may notice trouble with certain sites. Sometimes the robot never gets anywhere, stops after the first page, or misses a whole section of the site. This is usually because the Web page creators made links that are easy for humans using the Netscape Navigator or Microsoft Internet Explorer browsers to follow, but are somehow invisible to the robots.

II. How Robots Follow Links to Find Pages

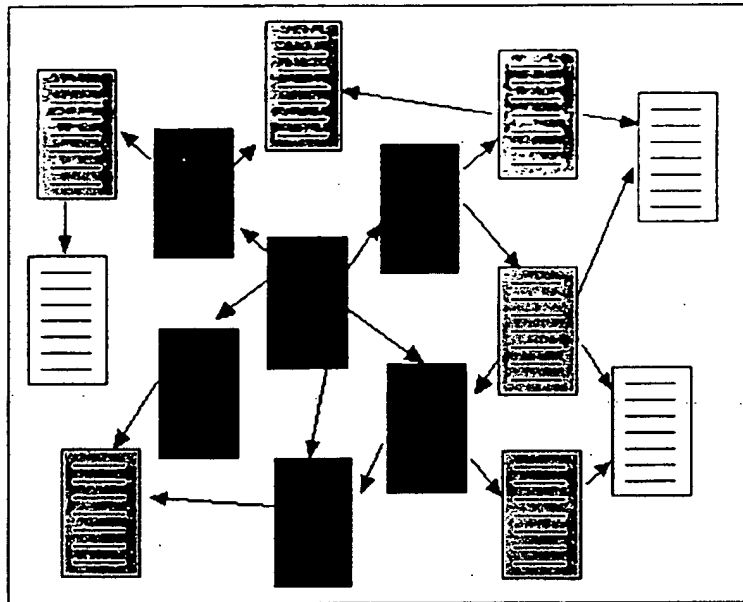
Robots start at a given page, usually the main page of a site, read the text of the page just like a Web browser, and follow the links to other pages. If you think of a site as a Web, the robot starts in the center, and follows links from strand to strand until it has reached every one. Some robots will also follow links to other sites, while most stay in the same server host name (such as `www.example.com`) or domain name (`example.com`), depending on your search administration settings.

Breadth-First Crawling

The idea of breadth-first indexing is to retrieve all the pages around the starting point before following links further away from the start. This is the most common way that robots follow links. If your robot is indexing several hosts, this approach distributes the load quickly, so that no one server must respond constantly. It's also easier for robot writers to implement parallel processing for this system.

Ultraseek Server uses concurrent *threads* (simultaneous program paths) so it processes several pages in parallel.

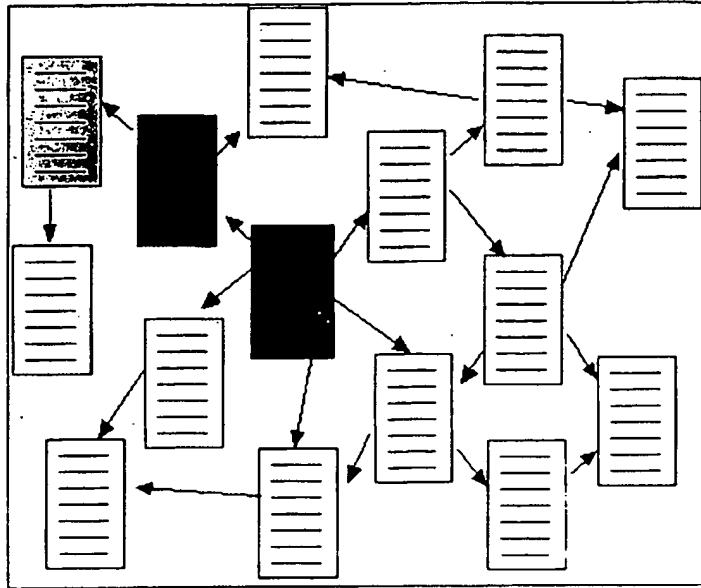
In this diagram, the starting point is at the center, with the darkest gray. The next pages, in the medium gray, will be indexed first, followed by those they link (in the light gray), and the outer links, in white.



Depth-First Crawling

The alternate approach, depth-first indexing, follows all the links from the first link on the starting page, then the first link on the second page, and so on. Once it has indexed the first link on each page, it goes on to the second and subsequent links, and follows them. Some unsophisticated robots use this method, as it can be easier to code.

In this diagram, the starting point is at the center, with the darkest gray. The first linked page is a dark gray, and the first links from that page are lighter grays.



Other Indexing Issues

Spidering Depth

Another issue with robots is how *deep* they will go into a site. In the example above of depth-first searching, the starting point is level 0, and the grays indicate an additional three levels of linking. For some sites, the most important information is near the starting point and the pages deep in the site are less relevant to the topic. For other sites, the first few levels contain mainly lists of links, and the detailed content is deeper in. In the second case, make sure the robot will index the detailed pages, as they are valuable to those who come to search the site. Some Web-wide robots will only index the top few levels of Web sites, to save space.

Ultraseek Server's spider lets you control this by limiting the number of directories in a URL (the number of slashes), which starts with 0 at the root directory and defaults to 10 directories. Ultraseek Server uses this depth count to avoid directory loops.

Server Load

Search engine robots, like Web browsers, may use multiple connections to read data from the Web server. However, this can overwhelm servers, forcing them to respond to robot requests to the detriment of human site visitors. When monitoring the server or analyzing logs, the Web administrator may notice many requests for pages from the same IP address. Many search engine robots allow administrators to *throttle down* the robot, so it does not request pages from the server at top speed. This is particularly important when the robot is limited to a single host and the server is busy with user interactions.

Overloaded servers, especially those with large files that do not change very often, or which pay per byte, may prefer robots that use the HTTP commands HEAD or CONDITIONAL GET. These commands allow a client, in this case the robot, to get meta information about a page, in particular, the date it was last modified. This means that the server only sends pages when they have been changed, rather than sending every page, when the robot traverses the site -- which can reduce the load on the server considerably.

Ultraseek Server's spider will only retain one outstanding request at a time from a server, although the default is to have up to five simultaneous threads indexing five separate servers. The spider can also slow down to reduce the server load: in the Network panel you can define the URLs and the number of seconds to delay.

The spider uses HTTP commands to check whether the file has been changed since the last time it was requested (this is known as "If-Modified-Since" and may show up in server logs as "CONDITIONAL GET". Using this system means that the server only sends the entire page when it has been changed, rather than sending every page when the robot traverses the site, and can reduce the load on the server considerably. For more information, see the HTTP 1.1 Protocol, section 14.25.

Robots and Private Data

Password-Protected Pages

If a site has a private, password-protected section, the robot may not be able to index the data stored there. In most cases, that is the right behavior, as a search on private data could reveal damaging or embarrassing information. In other situations, you may want to allow the robot access to the private section. Some local search engine robots, such as those used for sites or intranets, can store user names and passwords, and index those private sections.

Ultraseek Server's administrative interface allows you to specify a password for the spider to use when accessing each protected area. For example, you may want to make a special collection for planning documents, and only allow users who have access to those documents to search them. On the other hand, if you are selling reports, you might want to let prospective customers search your documents, to locate the most useful ones before purchase.

Encrypted Data

The SSL (Secure Sockets Layer) HTTPS protocol ensures that private data is encrypted and authenticated during transit through the Web. In addition to uploading credit card numbers, Web servers can use this system to encrypt banking information, patient records, student grades, and

other private data that should be kept confidential. Browsers contain special code to check the authenticity of a page and decrypt the text.

To index and search the private data, a search indexing robot must either include the SSL decryption code, access the pages without encryption (which can cause security problems), use a backend database rather than a search engine, or index using the file system rather than a robot. Note that the data is not encrypted in the search engine index, so search administrators should treat the index as sensitive and confidential, and implement stringent security measures.

Ultraseek Server offers SSL Spidering as an add-on. With this option, the spider includes SSL code, so no data will be transferred without encryption. When a searcher clicks on a link to a protected page in the search results, the code in their browser continues to protect the data as it is transferred.

III. Communicating With Robots

You'd think it would be hard to communicate with robots -- they are computer programs written by someone else, after all. But the designers of the robots have set up some ingenious ways for Webmasters to identify robots, track them and tell them where they are welcome or unwanted.

User Agents in Web Server Logs

Search engine robots will identify themselves when they request a page from a server. The Web HTTP communication protocol includes some information called a *header*, which contains data about the capabilities of the client (a Web browser or robot program), what kinds of information the request is for, and so on. One of these header fields is *User-Agent*, which stores the name of the *client* (the program requesting the information), whether it's a browser or a robot. If the server monitor or log system stores this information, the Web server administrator can see what the robot has done.

For example, if the Infoseek Web-wide robot is crawling a site, the text InfoSeek Sidewinder will be in the log. If a Web administrator has a problem with this robot, they can look up the information and contact the search administrator. The best search engine indexers allow the search administrator to set the text of this data, so it includes contact information, such as an email address or a URL to a Web page with an explanation of the robot and a feedback form.

The default Ultraseek Server User Agent is "Ultraseek," but search administrators can add their contact Web site and email address (for all or individual sites) in the Networking Panel. Customizing this information improves communication with the administrators of the sites indexed, allowing them to contact the owner of the spider if they have any questions.

For listings of search engine user agents, see the SearchEngineWatch Spider Spotting Chart, which is mainly for Web-wide search engines such as Infoseek, AltaVista and HotBot, or the Web Robots Database and the BotWatch list which cover robots of all kinds, including email harvesters and download bots. For a Perl script to analyze bots on a local site, see BotWatch.

Robots.txt

Robots should also check a special file in the root of each server called `robots.txt`, which is, as you may guess, a plain text file (not HTML). `Robots.txt` implements the Robots Exclusion Protocol, which allows the Web site administrator to define what parts of the site are off-limits to specific robot user agent names. Web administrators can disallow access to `cgi`, private and temporary directories, for example, because they do not want pages in those areas indexed.

Ultrasseek Server allows for full robots.txt support according to the robots.txt standard. Administrators can customize the user agent to meet their needs.

There is only one `robots.txt` file per Web server, and it must be located in the root directory, for example <http://www.domain.com/robots.txt>.

The syntax of this file is obscure to most of us: it tells robots **not** to look at pages that have certain paths in their URLs (that's why it's called an "exclusion" protocol). Each section includes the name of the user agent (robot) and the paths it may not follow. There is no way to allow a specific directory, or to specify a kind of file.

A robot may reasonably access any directory path in a URL that is not explicitly disallowed in this file.

This protocol is a form of communication between Web site administrators and robots; it does nothing to **prevent** a robot from accessing a file. There are other ways in which certain domains can be excluded from sites, if necessary. Your search indexing robot should honor these settings in most cases; contact the Webmaster for changes. The only exception is if an unresponsive Webmaster excludes all robots from the entire domain, and the person who is responsible for part of that domain would like to be indexed.

For more information, see the Standard for Robot Exclusion and the Web Server Administrator's Guide to the Robots Exclusion Protocol. You can also check for errors with the online `robots.txt` syntax checker.

For an explanation please see Exhibit 1.

Robots META Tag

In addition to server-wide robot control, Web page creators can also specify that individual pages should not be indexed by search engine robots, or that the links on the page should not be followed by robots. This is also useful where a page author does not have permission to change the `robots.txt` file.

A robots META tag, placed in the HTML `<HEAD>` section of a page, can specify either or both of these actions. Many, but not all, search engine robots will recognize this tag and follow the rules for each page. If certain pages are not indexed, check their headers to see if these tags are included.

Ultrasseek Server's spider recognizes the Robots META tag and follows the rules for each page.

The robot will still read the page, at least as far as the META tag, and the Web server log will show that the page was sent to the robot, even if it was not indexed. And, as with the robots.txt file, this is a form of communication between a page creator and a cooperating robot rather than a way to truly keep a page private.

For Robots META tag syntax, please see Exhibit 2.

Note: if you add Robot META tags to a framed site, be sure to include them on both the FRAMESET and the FRAME pages.

For more information, see the HTML Author's Guide to the Robots META tag.

IV. Indexing Process

This paper has been talking about robots following links and reading pages, but there's another step involved. *Indexing* is the process of extracting useful data from the source, in this case a Web page, and storing it in a file for later retrieval by the search engine.

Indexing starts by extracting individual words from the text of a page. Simple search indexers just read every word that's not in a tag, while others, such as the Ultraseek Server indexer, look for words by using sophisticated algorithms, so even those with punctuation, such as RJ-11, e-mail and AS/400, are searchable. They may also store text from useful HTML tags, such as the META Keywords and META Description tags, ALT tags (text related to images), and URL text. The indexer will also store the page URL and title, the Meta Description text (if any) so that it can display it in the results, and additional metadata, such as the author and date, especially when it's kept in a standard tag set such as the Dublin Core.

Ultraseek Server supports the Dublin Core set of META Tags and provides fielded searching using those tags.

Tracking the Robot

There are several ways you can follow the path of your search indexing robot, to understand what links it is following and which pages it is indexing and why. This can help you explain to site administrators and page creators why their data is not found during searches.

Site Map or Page Listing

Some search indexers generate a listing of the folders and files on the site (which they may call a site map). You can compare that to your understanding of the organization of the site, based on your navigation around the site, and identify sections that did not get indexed.

To see a listing of the pages located by Ultraseek Server's spider, select the View Sites button on the Collection Status panel of the admin. The list page shows the sites in the collection, with the number of URLs in the current queue, the number completed, and the number of documents (pages and other files) indexed. Clicking on the link to a site will show you a listing of the pages on that site.

Robot Index Log

Many search indexing robots keep logs of their progress, tracking each page they include in the index. When you start indexing a new site, or find problems locating some pages on the site, you can look at the log to find out which links the robot successfully followed. Some provide a lot of detail, including notice of problems and duplicate pages found.

Ultraseek Server's spider keeps a log of its progress, tracking each page it includes in the index. To activate the log, check the box labeled "Log disallowed URLs", then click OK on the Collections Filters panel of the admin. An example of the log, along with explanations is attached as Exhibit 3.

The index log may be viewed two ways:

- o Collection Status panel View Log button, which will show you the recent spider activity for the current collection only.
- o Server Parameters panel View Log button, which will show you a list of the stored query, access and error logs (the default is to only store seven of each). The error log also includes successful index entries, and covers all collections on this instance of Ultraseek Server.

Web Server Logging

If you have trouble figuring out why a robot can't find certain pages, you can use the Web server logs to watch it and track its progress through a site. While it's not a single line through the site, and you can't tell which page directs a specific URL, it does help you understand what was found and what was not (a bad link usually has a STATUS code of 404).

In Exhibit 4, you can see the robot traversing the SearchTools site. First it gets the robots.txt file, to learn about the disallowed sections, then it starts at the main index page and gathers several URLs to index. When it comes back to a page that it has seen before, it only gets the data if the page has changed since the last time it tried to index the page. If any pages have been removed since the last time the robot requested them, the error code is 404 and the indexer should remove that page information from the index.

V. Details of Following Links and Crawling Sites

For many sites, following links is easy and the robot spider logic is quite simple. So what is a link? A link is a legitimate URL, usually in the form `` (for a file in the same directory on the same server) or `` (for files on other servers), and so on. Seems fairly straightforward, so even a dumb robot can handle these links. (For a good introduction, see the Anatomy of an HTTP URL at WebReference.com and for technical information, see A Guide to URLs.)

However, some sites are hard to crawl. JavaScript, frames, image maps and other features added to HTML will confuse robots which can only follow links in text areas of the page. When you encounter a site which your robot can't crawl, investigation will reveal which of the many techniques the Web page creator has used to implement linking. This section will help you identify the code and offer suggestions for improvements.

Note: Visually-impaired people, and the browsers designed to help them surf the Web, have many of the same problems locating links, and need to follow links from pages rather than refer to a site map page. When sites are designed with accessibility in mind, robot spiders benefit as well. For more information, see the W3C Web Content Accessibility Guidelines. To test pages and locate problem areas, you can use the accessibility analyzer Bobby.

The Site Map Solution and its Limitations

One solution for these problems is to create and maintain a site map or page list, with links to all pages on your site. Then you could just use that as the starting page, and have the robot follow all links from there. There are three main problems with this plan:

- The larger the site, the more complex and confusing the site map page will become. This is somewhat of a problem with robot starting pages, but a more serious problem for humans trying to understand the site. Most useful maps for site visitors are carefully designed to convey information and organization using text, colors, shapes and/or spatial relationships. They do not link to every page or attempt to be complete, because that would be too confusing.
- It is extremely difficult to keep track of all changes on a site. In most cases, it will be easier for content creators and site designers to make robot-readable links on their pages, rather than trying to update the main site map page every time they add or change a link.
- To generate the site map automatically, a software program can gather the pages by following links or traversing folders. If it tries to follow links, it will have the same difficulties as a search robot indexer. If it uses the local file system to traverse folders, the administrator must be constantly careful about not adding files before they are public, removing obsolete files and excluding secure areas. These are difficult tasks and require a great deal of attention.

If possible, request that Web page creators provide alternate navigation on the same page whenever they use one of the special navigation systems described below.

Image Maps

Image maps are graphic images with links or actions associated with sections of the image. For example, a map of the US would have different links when you click on California, Illinois or Maine. There are two kinds of image maps, Client-Side and Server-Side:

Client-Side Image Maps

These are graphic files with associated information in the HTML page, including links. Some robot spiders can follow links in these formats, because they include the familiar HREF codes (without the "A" anchor tag).

Here is a simple image with client-side image map codes:



The associated map commands look like this:

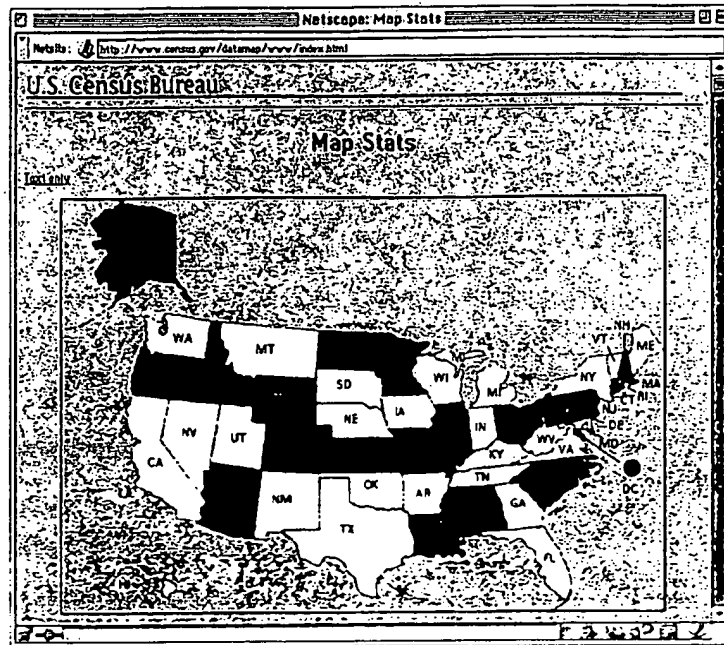
```
<map name="client-side-map"> <area shape="circle" coords="91,49,38" href="client-map-circle.html"> <area shape="rect" coords="180,19,293,78" href="client-map-rect.html">
</map>
```

Ultrasseek Server's spider can follow the links in client-side image maps, because they include the familiar HREF codes (without the "A" anchor tag).

Server-Side Image Maps

These maps store their links on the server and wait for the user to click on a particular location in the image before serving the link. Robots can't click on the map, so they are stuck. To serve robots (and non-graphic browsers), any site with server-side image maps should also have text links on the map pages, or a link to a page that lists all the image map links.

For example, the U.S. Government Census map has a nice link to a text listing of the states, next to the image map itself. Using this link, robots can access the information; if it wasn't there, they would be unable to get to any of the state pages.



The code itself is very simple, with the ISMAP attribute telling browsers that the image should be treated as an image map.

Robots and non-graphical browsers can't click on the map, so it's a good thing that the listing on the map.txt page is an alphabetical list of states. This allows robots to follow the links to the state pages and index the text on those pages.

For more information, see the W3C HTML 4 Specification for Image Maps.

Frames and Framesets

Frames are widely used for Web site navigation, but they are not easy for robots to navigate. In most cases, the first page loaded is the *frameset* page, which links all the pages together. Many automated and graphical HTML authoring tools do this automatically, so page creators may not even know that this frameset page exists! Robots do know, and they need some help navigating frames.

Some robots will simply not follow links inside the <FRAMESET> tag. Web page creators should always include links within the <NOFRAMES> tag section to provide alternate access. If you are running a robot like that, and there are no helpful links in NOFRAMES, try to locate a site map or text listing of links for this site.

Ultrasseek Server's spider follows links in FRAMESET pages and indexes those pages.

In addition to problems locating framed pages, both Web-wide and local search engines display those pages individually, rather than loading the frameset first. Many of these pages were never

meant to be seen separately, so they don't have any navigation features. This puts the searcher into a dead end, with no way to find the home page of the site, much less related pages. Search administrators should encourage all content creators to add links to the home page and, if possible, to the parent page or frameset page.

Exhibit 5 is an example of a page that has a FRAMESET of three frames (main, right and bottom), and a NOFRAMES section with a link to the main page and links to other useful pages. Any non-graphical or non-frame browser will be able to see navigate the site and locate the information it contains.

JavaScript Pages

JavaScript is a programming language that allows page creators to manipulate elements of Web browsers. For example, JavaScript can open a new window, switch images when a mouse passes over a button to produce a "rollover" effect, validate data in forms, and so on. JavaScript can also create HTML text within a browser window, and that can cause many problems for robot spidering and indexing text.

JavaScript HTML Generation

JavaScript can write text in HTML files, using the `document.write` or `document.writeln` command. Unless a robot contains a JavaScript engine, it's hard for them to recognize the links within these scripts. A robot can scan the script looking for "A HREF", ".htm", ".html", and "http://", but few do. In addition, many scripts create URLs dynamically, putting elements together on the fly, and cannot be detected programmatically unless a client includes a full JavaScript interpreter.



Ultrasseek Server does not include a JavaScript interpreter, so it cannot recognize or follow these links; however, the spider will follow links inside <NOSCRIPT> tags.

Exhibit 6 is a simple JavaScript example. It displays two different pieces of text, depending on whether the JavaScript interpreter is available or not. JavaScript browsers process the commands in the SCRIPT tag, while older browsers ignore all text in the HTML comment tag, but they display text in the <NOSCRIPT> tag. This allows page creators to set up an alternate display, and allows non-JavaScript browsers and robots access to the text and links.

JavaScript Menus and Lists

Many sites use JavaScript menus and scrolling lists as navigation systems. This allows users to select an item and go to the associated page. Unfortunately, most robots (and browsers without JavaScript) can't easily follow these paths, because the browser's scripting system puts together the "href" and the file path after a user has selected an item. While a robot could just try all items starting with "http://" or ending with ".htm" or ".html", most do not. In addition, some of these menus are built by on the fly by scripts that use customer IDs and other special codes, which are not available to robots.

Again, the solution is to request that the Web page creators use the <NOSCRIPT> tag, duplicate their popup menus and scrolling lists as bullet lists, and thus allow the robots to follow the links.

Redirecting Links

Redirect links provide a path from an old or inappropriate URL to a working URL. Redirects are a great thing for Web site creators, because they do not break bookmarks, old search engine indexes, or other links. When a request comes in for an old page, the redirect code tells the browser to get the new URL instead. However, they can cause problems for robots following these links.

Server Redirect Files

Many Web servers recognize a standard redirect format, although they may require a special file type or setting. This includes an HTTP version line and another line with the target URL in the standard format. For example:

```
HTTP/1.1 302
Found Location: http://www.domain.com/new/specials.html
```

When the server gets a request for the original page, it locates the redirect file and works with the browser or other HTTP client to send the target page defined in the redirect. Most browsers and robots accept the target in place of the original URL, and most search indexers will store the target as the URL in the index. However, some robots may be storing the original URL as an empty file, or match the contents of the target page with the original URL. If the redirect file is ever removed, the search engine may lose track of the target file.

Ultrasseek Server's spider correctly accepts the target in place of the original URL, storing the target as the document in the index.

To be sure the robot will always find redirected target files, the Web site maintainer should keep a list of these files on another page or a site map, so there's an alternate path.

META Refresh Redirection

Some servers do not provide access to redirect files; in this case, page creators can use a META tag to work with browsers or other HTTP clients to move to the preferred URL. The META tag is **http-equiv="Refresh"**, meaning that it acts as an HTTP command, rather than as plain text. The format is:

```
<META http-equiv="Refresh" content="10; URL=target.html">
```

The **content** parameter is the delay interval, the number of seconds that the browser should display the current page before moving to the target. The **URL** parameter is the path of the target page. Many Web page creators will set the delay to 10 seconds or more, allowing users to see the message that the page has been moved before bringing up the new page. This makes it more likely that they will update bookmarks and other links.

Ultraseek Server's spider follows the redirect link, but also reads the original page and follows the links there.

The text on this page should also include a normal link to the target page. Many old browsers and robots do not recognize the META Refresh URL and will not follow that link, so an HREF link in the text will allow them to find the target page. As they do not recognize the META Refresh tag, this technique will work even with a refresh delay interval of 0.

Tip: In most cases, page creators will want links followed, but they will not want this page indexed, so they should use `<META name="ROBOTS" content="NOINDEX">`.

Other Web Interfaces

Other Web interfaces, such as Java, and plug-ins such as Macromedia Flash, do not generate HTML Web pages. They are not part of the browser, although the page creator might not be aware of that. If you encounter one of these pages, request that the page creator provide an alternate HTML version. Some applications, such as Macromedia's AfterShock, will do this automatically (if the page creator uses the options to list all text and URLs).

Future standards, such as the W3C's Scalable Vector Graphics (SVG) specification, will provide standard formats for software to generate graphics from text, which will preserve the text. This means that non-graphical browsers, including robots, will be able to access the information and follow the links. Until that time, they will need plain HTML versions of the pages. Another important standard is XML (eXtensible Markup Language), which will have its own more powerful linking format called XLink. As these standards are growing, robots should be prepared to follow these links and to recognize and index XML-tagged text. For more information, see XML and Search.

Ultraseek Server 3.1 provides XML field searching and XML namespace support.

The Web used to have a simple interface: just HTML and graphics, as rendered by a browser. These formats are too simple for many of the interactive and graphical features that programmers have wanted, so they have invented other ways of interacting with end-users, within the context of the Web browser.

Graphic Text

Some designers like to control the look of their text on the page and generate GIFs or JPEG files with text. Although a human can read these words, a robot cannot read or index them.

While Ultraseek Server does not actually index the graphics, the spider reads the ALT tags attached to the graphics and enters them into the index for that document.

Plug-ins

Web designers use Plug-Ins to add features that are not available in the browsers. The most important of these are the **Shockwave** and **Flash** animation programs from Macromedia. These display within the browser window, so end users and even designers may not be aware that they are not HTML and are therefore invisible to robot spiders.

Macromedia provides a utility, **AfterShock**, which will generate HTML pages and links, if the designer uses the options to list all text and URLs. In this case, the search engine robots can index the pages.

Acrobat

Adobe Acrobat displays documents in a printable format with all layout and design intact, either as a standalone application or as a browser plug-in. Most of these documents were generated from word processing or layout programs, and they contain both the visual interface and the document text, while others were created by scanning, so they don't have any machine-readable text. Some search engine robots can follow links to PDF files, reading and indexing any available text, while others cannot.

Ultraseek Servers spider follows links to PDF files, then reads and indexes any available text.

Java

Java applets are supported by the major browsers, and can display within the browser windows, but again, they are invisible to robots. In general, these applets should be used only for optional interactive features, such as mortgage rate calculation. Unfortunately, some sites use Java to generate site maps, which are unusable by robot indexers.

Java applications on the server, such as those written to the Java Servlet API, can generate straightforward HTML pages, which are perfectly compatible with Ultraseek Servers spider.

ActiveX

Microsoft Internet Explorer browsers have an additional interface, connecting the browser to the Windows operating system. ActiveX interacts directly with the end user and sends back information to the server. None of this data is accessible to any Web robot. Any site using this format should also have a simple HTML version as well, for cross-platform compatibility, so the robot should be able to read and follow links in that version.

Non-Text File Formats

Some Web sites serve files in formats other than HTML and text, with file name extensions such as ".doc" (Microsoft Word), ".xls" (Microsoft Excel), and so on. When a site visitor clicks on a link to one of these pages, the browser downloads the file to the local hard drive, then works with the operating system to launch an application to read that file. Some search indexing robots have

compatibility modules that can translate those formats to text, so they can index the files. In that case, it will follow links with the operative file name extension, receive the file information, convert the text, and store the data in the index.

Ultraseek Server licenses technology from Inso that converts such non-text file formats as MS Word .doc and MS Excel .xls (among many others) into an index-capable format. Content from these documents is then stored in the index.

New Standard File Formats

The Web community, aware of the limits of HTML, is working on nonproprietary solutions for presenting information. Because these formats are open standards, and because the formats are designed to separate the text from the display, future versions of robots will be able to follow links and index text in very complex pages.

The most important of these standards is **XML** (eXtensible Markup Language), which will have its own powerful linking format called XLink. As these standards are growing, robots should be prepared to follow these links and to recognize and index XML-tagged text. A few search indexers can already recognize XML documents, and store some of the XML fields for separate searching.

Ultraseek Server recognizes XML documents, and the XML Mappings panel of the server settings allows search administrators to set up searches by field. For example, if you have a large set of book reviews, you can set up fields that were never in HTML documents, such as "Book Title", "Series", "Publisher", and "Reviewer", and allow searchers to specify which field they want.

Another standard, Scalable Vector Graphics (SVG) will provide standard ways for software to generate graphics from text, instead of using a proprietary text animation system. This means that non-graphical browsers, including robots, will be able to access the information and follow the links.

Complex URLs

Robot indexers often have trouble with URLs that have unusual file name extensions or characters in the path.

Files with Extensions Other than .html and .htm

Some pages are text, while others are created by server applications which generate HTML files, rather than being typed and saved onto disk. These files sometimes retain the original file extension, rather than the more usual ".html" and ".htm" extensions.

Examples of these include:

- no extension - HTML files formatted correctly by the server but missing the extension entirely.
- .txt - plain text files without HTML markup
- .ssi - HTML files containing text inserted by Server-Side Includes.
- .shtml - HTML files containing text inserted by Server-Side Includes.
- .pl - HTML files generated by Perl on the server.
- .cfm - HTML files generated by Allaire Cold Fusion on the server.
- .asp - HTML files generated by Microsoft Active Server Pages on the server.
- .lasso - HTML files generated by the Lasso Web Application Server.
- .xml - XML (eXtensible Markup Language) text files with special content-oriented tags rather than HTML tags

Some robot indexers will follow links to these files and attempt to index them as HTML or plain text, while others will ignore the files entirely. If your robot encounters one of these pages, it may fail to follow the link and therefore leave one or more pages not indexed. In that case, the page creator should change the options on their system or work with the server administrator to generate files with the .html extension.

Ultrasseek Server's spider follows links to these files using both the file name extension and the server's MIME type to recognize the text and HTML.

Note that some of the pages are created dynamically. For those cases, see the Locating Dynamic Data section.

Punctuation in URLs

Robot indexers follow links when the URLs are set up using the standard punctuation, as defined in the W3C Universal Resource Identifiers in WWW documents. These rules allow file names to include letters, numbers, . (period), ~ (tilde), - (dash), and _ (underscore).

Robots should also handle file paths with the delimiters used in standard URLs:

- / (Slash, delimiting directory hierarchy)
- . (Period, delimiting host name elements)
- : (Colon, delimiting port numbers)
- # (Hash or Pound Sign, delimiting anchor links)
- % (Percent Sign, for encoding spaces and other special characters)

All robots should be able to follow links with these characters in the URLs, and should never have any trouble with them. For more information, see the standard at RFC1808: Relative Uniform Resource Locators.

Ultraseek Server's spider finds documents conforming to the W3C document defined above, along with RFC 2396, including documents in file paths containing the described delimiters.

Some URLs, usually those involved with dynamic data, have other punctuation marks. These are:

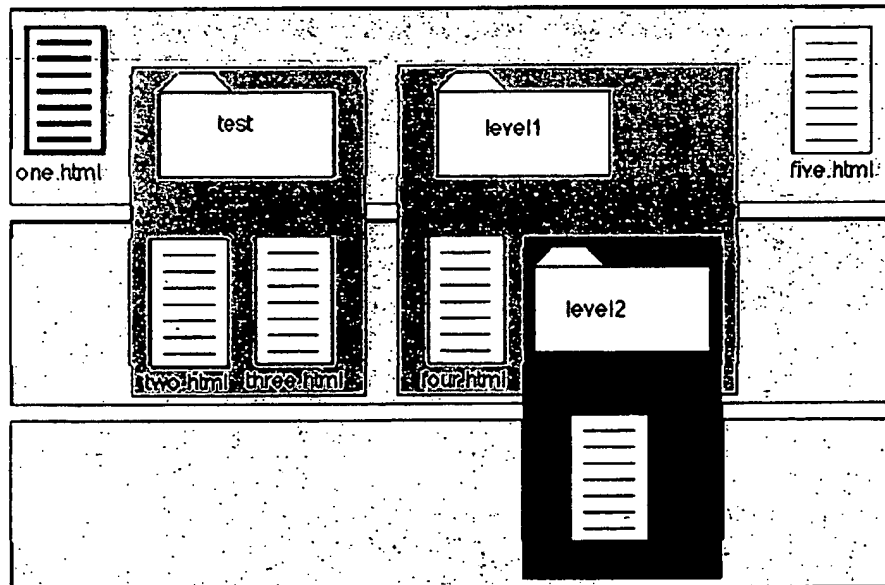
- ? (Question Mark or Query): defined by the HTML standard for sending form data to the server using the HTTP Get function
- \$ (Dollar Sign): often used for sending form data
- = (Equals Sign): used in the same standard for control name/value pairs
- & (Ampersand): used in the same standard for separating name/value pairs
- ; (Semicolon): also used for separating parameters

Unchecking the "Disallow all URLs with query strings" checkbox in Ultraseek Server's Collections Filters panel will allow the spider to follow links with these punctuation marks, indexing the content. Maintaining the default checked in this checkbox tells the spider to ignore any URLs with a question mark (?). There is no other punctuation limit in the spider.

Relative Links

Relative links are links to other files on the server without the full http:// URL format. They follow the standard Unix rules for relative file paths, and robots resolve them by comparing the current URL to the path in the link and adjusting as necessary.

For example, if you have a Web directory that looks like this:



Relative URLs would look like this:

type	originating file	link	points to
name followed by a / (slash) and a file name: start at this level, go into the directory, and open the document	/level1/four.html		/level1/level2/six.html
/ (slash): start at the top level of this host	/test/two.html		/five.html
../ (dot, dot, slash): start at the next level up, in the parent directory	/level1/level2/six.html		/level1/four.html
./ (dot slash) start at the current directory (this may be a typo)	/test/two.html		/test/three.html

Relative Link Problems

Robots encounter many relative link problems, often due to misunderstandings of how relative file paths should work. For example, some links include more parent directories (../) than are included in the current file path, so that the link points at a file that is not in the Web directory tree. Others use the special "." and ".." symbols incorrectly, or when they are not necessary.

These are particularly prevalent in directory listings, where the server automatically displays a page containing links to all the files and subdirectories in the current location.

If your indexing robot misses some pages, malformed relative URLs may be the reason. Robots may attempt to resolve them, report errors, or silently ignore them. You can search for links to these pages, or use a local or Web-based link checker. They should be able to check the URLs in much the same way as your search indexing robot. Once you find the problems, inspect them closely, retype them, and perhaps even create an absolute URL (starting from /).

For more information, see RFC 2396, W3C Addressing: URLs, Partial Form and W3C RFC 1808: Relative Uniform Resource Locators.

VI. Dealing with Dynamic Data

Dynamic data is text and HTML generated by a server application, such as a database or content-management system. Sometimes, this data is inserted into a normal HTML page, and no browser, robot or other client will ever know where it came from. In other cases, the page is created in response to a user entry in a form, and may only be applicable to a specific person at a specific time, rather than a static page. For example, a sales person might look up their commission for the quarter-to-date, or a reporter might check the votes counted for a candidate. Neither of these pages is appropriate for search indexing, but dynamically-generated catalog entries and informational pages are. The trick is in telling the difference.

Some search engine robots, especially those of public Web search engines such as AltaVista, Infoseek and Hotbot, do not want to index dynamic pages, so they tend to ignore links with ? in the URL. A few of the site, intranet and portal search engines, including Ultraseek Server, can

handle these pages properly, allowing the search engine to index the pages generated by the database or automated system together with static pages.

Pages from Content-Management Systems

Content-Management systems provide Web site creation teams with automated tools to store text and pictures, and generate Web pages in a specific format and structure. Many large sites take advantage of these products to track and control their sites more effectively, but there are some implications for search indexing robots.

A screenshot of a warning message from the Ultraseek Server's spider. The text reads: "Ultraseek Server's spider follows these URLs when the 'Disallow all URLs with query strings' option is unchecked in the 'Collections/Filters' panel." The text is displayed in a monospaced font on a dark background with a light border.

Page Identifiers

Content-management systems often use URLs with codes to identify specific pages. For example, the articles on the SFGate site have URLs that look like this:

```
http://www.sfgate.com/cgi-bin/article.cgi?  
file=/chronicle/archive/1998/10/20/BU21765.DTL
```

In this case, the ? starts the query information and the = provides a path to the file. For this site, you would want your robot indexer to recognize the question mark and equal sign, so it can locate the pages.

Entry Ids

Some sites like to know where you came from, so they can keep track of user movement from one section of a site to another. For example, the WebMonkey site uses "tw=" to figure out if you followed a link from a specific place, in this case, the home page:

```
http://www.hotwired.com/Webmonkey/99/36/index3a.html?tw=frontdoor
```

and if you come in from the archive of server-related topics, it will be:

```
http://www.hotwired.com/Webmonkey/99/36/index3a.html?tw=backend
```

They are really the same page, but the URL is slightly different, so you have to make sure you get rid of duplicate entries.

Session Ids

Some content-management systems generate a *session id* for identifying a specific user, instead of using a cookie. For example, the Sun Java Web Server can generate a unique number for each user and add it to the end of the URLs, so they look like this:

```
www.example.com/store/catalog;$sessionid$DA32242SSGE2
```

However, a search indexing robot may traverse the server in apparently-random order, and may pause between requests to avoid overloading the server. In that case, the server would add a

different session ID to some requests, so the URLs would be different. A robot that does not strip the ending section could continue to index endlessly, because there would always be links to "new" pages (those with different session IDs in the URLs).

However, a spider that does try to interpret the URL and strips a suspected session ID, may not be able to access the content either. A Web server that provides a session ID is likely to have session-based content. Session-based content may not be something that should be spidered -- order forms, preference setting pages, mailing list subscriptions, whatever.

Sun's Java Web Server (and their JSP engine) assume that a servlet/JSP page needs a session. The servlet or JSP code must explicitly turn off sessions. So session IDs may be present on pages that don't really require them. Web servers should not place session IDs on stateless content, and should work with the search engine admin to decide what content should be searchable, and how to make that content available to the search engine.

Ultraseek Server's spider does not try to strip Java Web Server session IDs from URLs. If these URLs cause a problem for the spider, a filter disallowing them can be added:

```
Disallow: /*sessionid$
```

If there are pages with gratuitous session IDs, these can be accessed directly without a session ID. The site could be listed under an "allow text" filter, and the Webmaster could provide a page with links to all pages that should be indexed. Those links, without session IDs, would "seed" the spider with good URLs. The URL for that page should be added to the roots for the collection.

Cookies

Cookies are stored user identification in a format set up by browsers. They are generally required for shopping baskets and other interactive processes. Active Server Pages have cookies built in, but many never use them. A few search engines, including Ultraseek, can recognize stored cookie information.

In the Network panel of the Ultraseek Server admin, the "Additional HTTP Headers" section can include permanent cookies.

Lotus Notes Pages

The Lotus Notes Web server provides a very rich document structure, allowing site visitors to view the same information in many different ways. For example, the notes.net site allows you to view newsletter articles by author, date, language, level and product version, and to click

disclosure triangles to see links to the individual pages. Therefore, the main page of the archive has the URL:

<http://notes.net/today.nsf/Author?OpenView>

but each of the small triangles is an image link to another page, with another URL like this:

<http://notes.net/today.nsf/Author?OpenView&Start=1&Count=30&Expand=3#3>

which itself contains additional links to other pages. As you can imagine, this can send a robot into almost infinite gyrations, as it attempts to chase every combination of pages. If you are attempting to spider a Lotus Notes-driven site, consider using a robot that already knows about these kinds of sites.

checking the "Filter Lotus Domino navigation links" in the Collections Filters panel enables Ultraseek Server's spider to correctly handle Domino-driven sites.

Pages from Catalog-Style Databases

Database and middleware systems can generate Web pages in real time, formatting the content of records in the database into HTML. If the search indexing robot can traverse the links to these pages, it can incorporate the data into the index with the static pages, allowing site visitors to search both kinds of information at the same time, and see the results in relevance ranking order. Catalog-style databases are even more complex than content-management systems. For example, the URL for the Wonder Woman action figure page at Things from Another World looks like this (in its simple form):

<http://www.tfaw.com/Level4.html?SEC=HERE&SUBSEC=toys&ITEM=ARR1000000397>

The item number is key here: each item is unique and should be indexed separately. There are several approaches to having robots follow links for database and catalog data:

1. Generate static text pages from a database, store them in the file system, and create a page of links to these pages. Every time the database or catalog changes, the static pages must be regenerated.
2. Have the database itself generate pages that use a standard format so the robot cannot tell that the data is generated on the fly. Instead of the URL www.example.com/products?name=bluewidget:level=3, you could set it to www.example.com/products/bluewidget/3.html. That is what Amazon does, for example, generating plain URLs using the Taxis relational database system.
3. Choose a search indexing robot that can follow links to the dynamic data.

Unchecking the "Disallow all URLs with query strings" in the Collections Filters panel will tell Ultraseek Server's spider to follow these types of URLs.

When indexing dynamic data, robots should generally index the data itself, but not follow links onto listing pages. Robots should not index any directory or listing pages -- these pages do not

contain data that a searcher would want to locate. When these listing pages are generated, they should have META ROBOTS NOINDEX tags, so the robot can recognize their status.

Ultraseek Server by default ignores listings on these pages, but follows links except for parent links (i.e., the " " link to the parent directory). These settings can be changed in the Collections Filters panel.

For example, a mailing list archive should not allow data on the listing pages to be indexed, because the individual messages are the ones which present the actual data. In message pages themselves, robots should not follow "previous" and "next" links, as they are simply alternate ways of getting to other messages, so the page should be generated with the META ROBOTS NOFOLLOW setting.

Dynamic Web Applications and Black Holes

Another form of dynamic data is generated in real time by systems such as Web Calendars. These can automatically create new year pages and new day pages almost endlessly -- they do not expect people to follow the "Next year" link or to click on every day's link. Robots will continue to request these URLs, even five or ten years into the future, although there is no information to index. They can't tell that they are simply having a mechanical conversation with an automated system.

Some search engine robots can limit the number of *hops* from the Root URL. The limit reduces the likelihood that the robot will get into some other kind of endless loop, such as a long-term calendar or other dynamic data source, that may increment a date rather than add a directory layer. So the robot might follow links in the calendar for 100 months but no more.

In addition, a limit to the number of directories (delimited by slashes) avoids the situation where a file can accidentally link to itself, such as an alias, shortcut or symbolic link loop. For example, it might look like this: `www.domain.com/test/test/test/test/test/test/`. The default limit of 10 keeps this from going forever, although you can override it if the site is very deep.

Administrators can set the maximum numbers of hops from the root URL, along with the maximum number of directories in a URL on the Collection Filters panel of the Ultraseek Server admin.

VII. Detecting Duplicate Pages

A search indexing robot may encounter a page following links from several other pages, but it should only index it once. It must keep a list of the URLs it has encountered before, but in some cases the URL is different but the pages are the same. Search indexing systems can be programmed to recognize some of these cases, while others are only detectable by tracking pages carefully and comparing new pages to those already in the database.

Ultraseek Server's Collections Dupes panel provides sophisticated tools for defining duplicates and deciding which of the two or more pages to consider definitive. It allows you to define whether a page is a duplicate by comparing the title and summary or by completing a more thorough comparison of the entire page.

Default Index Pages

The HTTP protocol supports links to a directory, and allows servers to display a default page in that case. That's how we can link to a host home page without typing the page name every time, so `www.example.com` is the same as `www.example.com/index.html` (or `default.htm`, etc). Robot indexers should recognize this relationship, and only index each page once.

Ultraseek Server's spider recognizes this relationship and only indexes each page once.

Capitalization

Some Web servers are case-sensitive (they consider a directory or file name with capital letters as different from the same words with lowercase letters). For those servers, two pages with the same name but different cases (`example.html` vs. `Example.html`) might or might not be the same. The only way to tell is to compare the pages themselves.

Ultraseek Server compares the pages. The Collections Tuning panel also provides a checkbox to change the option from the default setting of ignoring page name text case, to account for those few sites where the pages are different.

Server Mirroring

If your robot is indexing several hosts in the same domain, one or more of them may mirror another. One may be named `www.example.com`, for example, and the other `www2.example.com`. DNS routers can automatically direct incoming HTTP request to one or the other to balance the server load. Therefore the indexing robot may locate a page on both servers, but they are truly duplicate pages. In this case, you really need a robot that can store the equivalence and convert all secondary host names to the primary host name before following links.

Ultraseek Server's spider will follow all links, even on different servers, but the Collection Dupes panel lets you prefer the main host when the indexer gets the same page from two hosts.

Redirects

Server redirects accept one URL but serve a different URL. These are useful for updating site structure, changing names and dealing with errors. If a site uses redirects, the robot should not store the original URL for the page, but that of the ultimate page served. Otherwise, the page could appear twice in the index, once for each URL.

Alternate Paths and Page Copies

In some sites, the content-management system can serve the same page in several categories. Although hypertext allows links from one category to another, the back end system may not be able to do the same, so the same page is created although the path to the page is different. On other sites, designers simply create copies of useful pages, rather than linking to the page in a different directory. This can result in duplicate pages, and in unsynchronized changes to these pages. A few search indexers, such as Ultraseek Server, will compare page contents and only index one copy.

Entry IDs

With Entry IDs, the same page can have multiple URLs with different links at the end. Any robot which follows these links will get a URL that is identical except for the final parameters, and which points at the same page. If an indexer doesn't recognize that they are the same page, it can't delete it from the index.

Ultraseek Server's Duplicate URL Preference lets you indicate that those with the Entry ID code at the end should be ignored if there is one without that ID.

VIII. Revisiting Sites & Updating Indexes

As sites change, a search engine robot must return periodically to update the index. Searchers justifiably hate to work with an index that does not match the current state of the data exactly, so the index updating system should be synchronized to the site content updates.

Update Schedules

The simplest updating scheme allows a search administrator or Webmaster to have the robot check for new data when they publish new content on the site. This simply tells the search indexing robot to start following links and reading pages again.

Ultraseek Server has a very powerful scheduling system. In addition to an on-demand index option, the Collection Tuning panel has a schedule for updating the index. Ultraseek also has an internal system that keeps track of how often each page is updated, and revisits the frequently-changed pages first. This is a very efficient solution because it concentrates on those sites or sections which change often, and spends less time checking those which are static.

Locating changed pages

As mentioned above, if the Web server reports the page modification date properly, some robots will only retrieve changed pages, and just get notification if the page was the same. However, some servers do not report the modification date correctly. Other robots just retrieve every page and re-index it, or compare it to the contents of the index and update if the page has changed.

Ultraseek Server's index update does not start from scratch, it modifies the index so you can continue to search while updating in the background. For the servers that do not properly report the modification date of files, Ultraseek Server can compare the pages and track when the content was actually changed. The tracking system reduces the number of times that the spider will check the server for relatively static data, and makes sure that changing data is indexed most often.

IX. Conclusion

Running a robot is always a challenge, so you want your search tool to make it as easy as possible. Search indexing robots should follow all the robot standards and allow you to control the speed of crawling. The right robot for you is the one that works properly on your sites, whether you have image maps, frames, JavaScript, complex URLs, relative links or dynamic data. It should provide helpful indexing reports, detect duplicate pages, and update according to your content change schedule. Use this article as a checklist to understand the strengths and weaknesses of your search engine robot, and how well it matches your sites.

Ultraseek Server's spider makes building an index as easy as possible. The browser administration lets you control the spider with filters and options, or define which duplicate file should be retained, and view the logs to track the spider's progress from any machine on your network. Even if you have a complex set of servers with a variety of formats, Ultraseek Server gives you the power to find and index all the pages you need.

X. Useful Robot Sites

- **W3C HTTP 1.1 Protocol, Section 14.25:**
<http://www.w3.org/Protocols/rfc2616/sec14.25>
- **Search Engine Watch Spider Spotting Chart:**
<http://searchenginewatch.com/Webmasters/spiderchart.html>
- **Robots Exclusion Protocol:**
<http://info.Webcrawler.com/mak/projects/robots/norobots.html>
- **Admin's Guide to the W3C Robot Exclusion Protocol:**
<http://info.Webcrawler.com/mak/projects/robots/exclusion-admin.html>
- **HTML Author's Guide to the Robots META Tag:**
<http://info.Webcrawler.com/mak/projects/robots/meta-user.html>
- **Dublin Core Metadata Initiative:** <http://purl.oclc.org/dc/>
- **Anatomy of an HTTP URL:** <http://www.Webreference.com/html/tutorial2/2.html>
- **A Guide to URLs:** <http://www.netspace.org/users/dwb/url-guide.html>
- **W3C Web Content Accessibility Guidelines:** <http://www.w3.org/WAI/GL/>
- **Bobby:** <http://www.cast.org/bobby/>
- **W3C HTML 4 Specification for Image Maps:**
<http://www.w3.org/TR/html401/struct/objects.html>
- **W3C HTML 4 Specification for Frames:**
<http://www.w3.org/TR/html401/present/frames.html>
- **W3C Scalable Vector Graphics specification:** <http://www.w3.org/TR/SVG/>
- **W3C's XML page:** <http://www.w3.org/XML/Activity>
- **W3C's Xlink page:** <http://www.w3.org/TR/xlink/>
- **XML and Search:** <http://www.searchtools.com/related/xml.html>
- **W3C Universal Resource Identifiers:** <http://www.w3.org/Addressing/URL/uri-spec.html>
- **RFC1808:** <http://www.w3.org/Addressing/rfc1808>
- **RFC2396:** <http://www.ics.uci.edu/pub/ietf/uri/rfc2396.txt>
- **Form Data Standards:** <http://www.w3.org/TR/1998/REC-html40-19980424/interact/forms.html#h-17.13.3>

- **W3C HTML 4.0 Specification, Appendix B, Notes on helping search engines index your Web site:** <http://www.w3.org/TR/PR-html40/appendix/notes.html>
- **Web Robots FAQ:** <http://info.Webcrawler.com/mak/projects/robots/faq.html>
- **Web Robots Database:** <http://info.Webcrawler.com/mak/projects/robots/active.html>
- **Guidelines for Robot Writers:**
- **UKOLN WebWatch Robots.txt checker:** <http://www.ukoln.ac.uk/Web-focus/Webwatch/services/robots-txt/>
- **BotWatch robots.txt syntax checker:** <http://www.tardis.ed.ac.uk/~sxw/robots/check/>
- **BotSpot:** <http://www.botspot.com/>

XI. Exhibits

- Exhibit 1: Robots.txt Syntax
- Exhibit 2: Robots META Tag Syntax
- Exhibit 3: Ultraseek Server Index Log Contents
- Exhibit 4: Sample Log File
- Exhibit 5: FRAMESET Example
- Exhibit 6: Sample Java Script Code

Exhibit 1: Robots.txt Syntax

Task	Entry	Notes
Allow robots complete access to the server	User-agent: * Disallow:	* means all user agents (robots). Because nothing is disallowed, everything is allowed.
Exclude all robots from part of the server	User-agent: * Disallow: /cgi-bin/ Disallow: /tmp/ Disallow: /private/	* means all user agents. The robots should not visit any pages in these directories.
Exclude a single robot	User-agent: BadBot Disallow: * User-agent: * Disallow: /priv/	In this case, the BadBot robot is not allowed to see anything. All other agents (*) can see everything. The blank line indicates a new "record" - a new user agent command.
Exclude a robot from a single file	User-agent: WeirdBot Disallow: /links/listing.html User-agent: * Disallow: /tmp/ Disallow: /private/	This keeps the WeirdBot from visiting the listing page in the links directory, while all other robots can see everything except the temp and private directories. The * for all other robots should always be at the end of the robots.txt file.

Exhibit 2: Robots META Tag Syntax

Task	Entry	Notes
Do not index, but follow links	<code><meta name="ROBOTS" content="NOINDEX"></code>	Use this for pages with many links on them, pages without not much useful data (such as a large image map); or for data that changes very frequently, such as news headlines. Because "follow" is the default, you don't have to include it.
Index, but do not follow links	<code><meta name="ROBOTS" content="NOFOLLOW"></code>	Use this for pages which have useful content but links which may be irrelevant or obsolete, or which may point to other sites.
Do not index or follow links	<code><meta name="ROBOTS" content="NOINDEX, NOFOLLOW"></code>	This is for pages which should not be indexed at all. If you put that in every page, the site should not be indexed.
Index and follow links	<code><meta name="ROBOTS" content="INDEX, FOLLOW"></code>	This is the default behavior: you don't have to include this tag.

Exhibit 3: Ultraseek Server Index Log Contents

The Ultraseek Server robot reports everything it encounters, if you have the "Log disallowed URLs" checked.

Simple pages without duplication look like this:

Sep 09 11:32:36 Info: [searcht] 200 OK:
<http://www.searchtools.com/info/guide.html>

Pages on other sites (which are not part of the collection) look like this:

Sep 09 11:32:35 Info: [searcht] 406 Disallowed by
URL filter: <http://perl.org/>

Pages on the same site which are not allowed by the filters look like this:

Aug 31 14:10:44 Info: [avi] 200 OK:
<http://www.hotwired.com/Webmonkey/>
Aug 31 14:10:44 Info: [avi] 406 Disallowed by URL
filter: <http://www.wired.com/home/copyright.html>

Duplicate pages look like this:

Sep 09 11:34:35 Info: [searcht] Duplicate of
<http://www.searchtools.com/info/conferences-past.html>:
<http://www.searchtools.com/info/conferences.html>

Pages not changed since the last check look like this:

Sep 09 11:34:52 Info: [searcht] 304 OK,
unchanged:
<http://www.searchtools.com/tools/overviews.html>

or

Sep 09 10:27:25 Info: [example] 304 Not modified:
<http://www.example.edu/fishing/localinfo.htm>

Pages changed since the last check look like this:

Aug 31 14:08:05 Info: [avi] Deleted previous
<http://www.hotwired.com/Webmonkey/kids/lessons/We b.html>
Aug 31 14:08:06 Info: [avi] 200 OK:
<http://www.hotwired.com/Webmonkey/kids/lessons/We b.html>

URLs which are the source of redirects:

Aug 31 11:38:09 Info: [avi] 302 Moved
Temporarily: <http://www.Webmonkey.com/>

Aug 31 14:11:06 Info: [avi] 301 Moved
Permanently:
<http://www.hotwired.com/Webmonkey/software>

URLs with parameters (text after question marks) are disallowed if not explicitly allowed, and the errors look like this:

Aug 31 14:10:44 Info: [avi] 406 Disallowed by URL
filter:
<http://www.hotwired.com/Webmonkey/xml/?tw=frontdo>
or

Excessively long URLs are reported as errors:

Aug 31 14:08:04 Info: [avi] 406 URL too long:
http://ads.example.com/event.ng/Type=click&ProfileID=8167&RunID=16250&AdID=21542&GroupID=1&FamilyID=3008&TagValues=2.5.6.25.156.159.174.322.374.389.411.73975.74235.74975.75378&Redirect=http:%2F%2FWebfarm.example.com%2Fclick_thru_request%2F164-1361k-1052%3Fr%3D1999.8.31.21.7.29.0

Pages with the META NOINDEX or NOFOLLOW tags are recognized:

Sep 09 11:35:29 Info: [searcht] 200 OK, nofollow
robots meta tag:
<http://www.searchtools.com/test/robots/meta-nofollow.html>

Sep 09 11:35:30 Info: [searcht] 200 OK, noindex
robots meta tag, nofollow robots meta tag:
<http://www.searchtools.com/test/robots/meta-noindex-nofollow.html>

Formats and MIME-types not recognized by the Ultraseek indexer:

Sep 09 16:15:42 Info: [searcht] 406 Unsupported
content-type application/octet-stream:
<http://test.searchtools.com/tools/tools.bin>

When the server does not respond quickly, the page is never received:

Aug 31 14:09:05 Info: [avi] 503 tsocket.timeout:
:

`http://www.hotwired.com/Webmonkey/guides/Web/nav_
page6.html`

When the server can't be found, the log includes an error (and keeps trying):

`503 socket.error: host not found:
http://nonexistant.example.com`

Exhibit 4: Sample Log File

In this log file, we have simplified the columns to the following information:

- CS-URI is the URL
- CS-STATUS is the result code (200 for good pages, 301 and 302 for redirects, 304 for unchanged, and 404 for not found)
- METHOD is the kind of request from the browser or robot, GET is the usual form for static pages, CONDITIONAL_GET indicates that the client only wants the page if it's changed since a specific date)
- AGENT is the User-Agent name of the robot (you can change this in Collection Network panel, and customize it for specific hosts)
- HOSTNAME is the server on which the Ultraseek server software is running
- BYTES_SENT is the amount of data transferred
- TRANSFER_TYPE is the time to send the data, in seconds

```
CS-URI CS-STATUS METHOD AGENT HOSTNAME BYTES_SENT  
TRANSFER_TIME  
/robots.txt 200 GET Ultraseek sunset.infoseek.com. 389 1  
/index.html 200 GET Ultraseek sunset.infoseek.com. 36998  
27  
/info/guide.html 200 GET Ultraseek sunset.infoseek.com.  
34904 17  
/tools/tools.html 200 GET Ultraseek sunset.infoseek.com.  
17702 2  
/info/news.html 200 CONDITIONAL_GET Ultraseek  
sunset.infoseek.com. 34847 21  
/surveys/ 302 GET Ultraseek sunset.infoseek.com 0 2  
/surveys/surveys.html 200 GET Ultraseek  
sunset.infoseek.com. 4218 16  
/index.html 304 CONDITIONAL_GET Ultraseek  
sunset.infoseek.com. 151 4  
/temp/surveys/index.html 404 GET Ultraseek  
sunset.infoseek.com. 2742 10
```

In particular, watching the 404 error entries tells you about bad link URLs, and site administrators should check these whenever possible.

Exhibit 5: Frameset Example

```
<FRAMESET cols="160,*" border=0>
  <FRAME src="http://www.domain.com/main.html" scrolling="auto"
border=10 NAME="main">
  <FRAME src="right.html" scrolling="auto" border=10
NAME="right">
  <FRAME src="bottom.html" scrolling="no" border=0 marginwidth=0
marginheight=0 NAME="bottom">
</FRAMESET>

<NOFRAMES>
  <BODY><H1>Welcome to our Site!</H1>
  <H2><A HREF="http://www.domain.com/main.html">Main
Page</A></H2>
  <H2>Other Pages</H2>
  <UL>
    <LI><A HREF="trees.html">Trees</A></LI>
    <LI><A HREF="flowers.html">Flowers</A></LI>
    <LI><A HREF="mammals.html">Mammals</A></LI>
    <LI><A HREF="insects.html">Insects</A></LI>
    <LI><A HREF="fish.html">Fish</A></LI>
  </UL>
  <P><A HREF="home.html">Home</A></P>
  <P><A HREF="about.html">About Us</A></P>
</BODY>
</NOFRAMES>
```

In this example, the FRAMESET will show the pages main.html, right.html and bottom.html, while the NOFRAMES will direct a user or robot to the main page or specific other pages. For the definitive reference information, see the HTML 4 specification for Frames.

Exhibit 6: Sample Java Script Code

```
<SCRIPT LANGUAGE="JavaScript">
  <!--
    document.writeln("<H3>JavaScript Example</H3>\n
    <P>This text is generated by a JavaScript. If you can see
it, you are using a \
JavaScript-compatible \
browser or other HTTP client that contains a JavaScript
interpreter. For \
more information, see <A
HREF='http://www.example.edu/javascript/'> \
our JavaScript Pages</A>. <\P> \
")
  // -->
</SCRIPT>

<NOSCRIPT>
  <P>This is only visible to browsers that cannot interpret
JavaScript.
  For more information,
  see the <A HREF='http://www.example.edu/javascript/'>
  our JavaScript Pages</A>. <\P>
</NOSCRIPT>
```

In the NOSCRIPT section of the code, the link is available for robots and non-JavaScript browsers.